Chiem Exercise 7

MAT 115

Exercise #7

Chapter 4 is about working with *tidy data*. When you want to learn something from data (eg, by visualizing and/or do statistical analysis), the usual first step after collecting the data is to wrangle data into tidy format. Refer back to the image from our first day of class: Data science proccess.

We will use the `tidyverse` package, so let's load that first. And we might as well load the `dslabs` package also.

```r
#install.packages('tidyverse')
library(tidyverse)
library(dslabs)
```

4.1 Section 4.1 explains what data in tidy format looks like: each row is one observation (or *case*) and each column represents one variable pertaining to that observation. See the book for some examples. You should think of each row as a `unit of analysis`.

Sometimes the distinction between tidy and not-tidy formats is not so clear cut. We can agree that the `murders` dataset is in tidy format, since each row is one case (in this circumstance, each row is a state) and each column in one variable. The unit of analysis is a state. Among others, we want to compare the states; it's not our aim to compare countries.

We can also agree that the `exams` dataset is in tidy format:

```r
load("exams.rda")
head(exams)
```

```
##   ID exam1 exam2 improve
## 1  1    48    48       0
## 2  2    47    46      -1
## 3  3    39    42       3
## 4  4    46    47       1
## 5  5    45    45       0
## 6  6    50    50       0
```

where each row is a student and the variables are exam 1 score, exam 2 score, and the improvement for each student. So this format is tidy, indeed.

Or is it? The data in `exams` can be formatted this way:

ID	Which Exam	Score
1	first	48
1	second	48
2	first	47

ID	Which Exam	Score
2	second	46
etc	etc	etc

Compare this format with the fertility rate example from the `gapminder` dataset in the book:

```r
# The following is code to generate the example in the book
# You can ignore the code for now, though it is a sneak peak into our future.

data("gapminder")
tidy_data <- gapminder %>%
  filter(country %in% c("South Korea", "Germany") & !is.na(fertility)) %>%
  select(country, year, fertility)
head(tidy_data, 6)
```

```
##       country year fertility
## 1     Germany 1960      2.41
## 2 South Korea 1960      6.16
## 3     Germany 1961      2.44
## 4 South Korea 1961      5.99
## 5     Germany 1962      2.47
## 6 South Korea 1962      5.79
```

The new format of the `exams` data is quite similar to the fertility example.

Can you explain in your own words how they are similar?

A: Both are formatted in rows and columns. Each column has a heading (with the data type) thats specifies what the column represents. The rows contain the number of each row from 1-n. The variables are also similary spaced apart. Both only have three headings.

So we can say that the new `exams` format is tidy. Each row represents one exam, and the variables are as indicated. In this format, the unit of analysis is an exam, not a student.

Which tidy format you should use depends on your unit of analysis and what your goals are. In future exercises we will explore some very powerful and useful functions for converting datasets between formats in order to make them tidy.

For the `exams` dataset, the main goal is to assign grades to students, so I think the first format is most useful.

Can you think of advantages of the second format? Answer here: It is more compact. It is convenient for looking at both exams because it is all in one column.

4.3 Section 4.3 is basically a tour of the `dplyr` part of `tidyverse`. The package `dplyr` ("dataframe plier") was created specifically to manipulate dataframes: adding variables, selecting parts by columns and rows, and sorting rows.

First up is the `mutate` function. It is used primarily to add columns to a dataframe. You can see an example in section 4.3.1. Here is another example. Suppose you want to calculate the average exam score for each student in the `exams` dataset. You actually know how to do this without using `mutate`, but let's use it anyway.

```
newexams <- mutate(exams, average=(exam1+exam2)/2)
head(newexams)
```

```
##   ID exam1 exam2 improve average
## 1  1    48    48       0    48.0
## 2  2    47    46      -1    46.5
## 3  3    39    42       3    40.5
## 4  4    46    47       1    46.5
## 5  5    45    45       0    45.0
## 6  6    50    50       0    50.0
```

Note that we do not have to use the $ symbol here. *Can you explain why?* A: We have already listed data fram in first argument

Suppose there is a third exam in the class. How do you add exam3 scores to the ***exams*** *dataframe? Answer here:*

A: Using mutate you would input it as the second argument.

Of course, you already know how to add a new column to a dataframe. How do you do it without the ***dplyr*** *package?*

A: dataframe$[newColumnName] <- [newColumnName]

```
# Try it out here.
#exams3 <- 1:19;

#exams$exams3 <- exams3

#exams
```

We use `mutate` to add columns to a dataframe. We can use `select` to delete columns. For example, suppose we want to delete the `improve` column from the `newexams` dataframe.

```
select(newexams,-improve)
```

```
##    ID exam1 exam2 average
## 1   1    48    48    48.0
## 2   2    47    46    46.5
## 3   3    39    42    40.5
## 4   4    46    47    46.5
## 5   5    45    45    45.0
## 6   6    50    50    50.0
## 7   7    50    45    47.5
## 8   8    41    43    42.0
## 9   9    50    46    48.0
## 10 10    47    46    46.5
## 11 11    41    39    40.0
## 12 12    50    50    50.0
## 13 13    47    47    47.0
## 14 14    47    45    46.0
## 15 15    47    31    39.0
## 16 16    42    31    36.5
## 17 17    47    49    48.0
## 18 18    46    45    45.5
## 19 19    46    39    42.5
```

Note the minus sign in front of the `improve` variable name. This tells the `select` function to delete that variable. What happens if we do not use the minus sign? Can you do this same thing using base R functionality?

A: It will remove everything except improve.

```
# Try it out here.
#exams$exams3 <- NULL
#exams
# Try also having more than one column included.
```

So the `select()` function basically chooses which columns to include. Using a minus sign signifies the opposite of inclusion. For rows, we use the `filter()` function. Note that selecting certain rows basically chooses a subset of your data.

For example, suppose we want to see which students have a perfect exam average.

```
filter(newexams,average==50)
```

```
##   ID exam1 exam2 improve average
## 1  6    50    50       0      50
## 2 12    50    50       0      50
```

Or we want to see which students do better than the mean in exam 2:

```
filter(newexams,exam2 > mean(exam2))
```

```
##    ID exam1 exam2 improve average
## 1   1    48    48       0    48.0
## 2   2    47    46      -1    46.5
## 3   4    46    47       1    46.5
## 4   5    45    45       0    45.0
## 5   6    50    50       0    50.0
## 6   7    50    45      -5    47.5
## 7   9    50    46      -4    48.0
## 8  10    47    46      -1    46.5
## 9  12    50    50       0    50.0
## 10 13    47    47       0    47.0
## 11 14    47    45      -2    46.0
## 12 17    47    49       2    48.0
## 13 18    46    45      -1    45.5
```

Now it's your turn. How do you select just those students whose scores on exam 1 and exam 2 are different?

```
# Write your code here.
filter(newexams,exam1 != exam2)
```

```
##    ID exam1 exam2 improve average
## 1   2    47    46      -1    46.5
## 2   3    39    42       3    40.5
## 3   4    46    47       1    46.5
## 4   7    50    45      -5    47.5
## 5   8    41    43       2    42.0
## 6   9    50    46      -4    48.0
## 7  10    47    46      -1    46.5
## 8  11    41    39      -2    40.0
## 9  14    47    45      -2    46.0
## 10 15    47    31     -16    39.0
## 11 16    42    31     -11    36.5
## 12 17    47    49       2    48.0
## 13 18    46    45      -1    45.5
## 14 19    46    39      -7    42.5
```

```r
# There are several ways to do this.
```

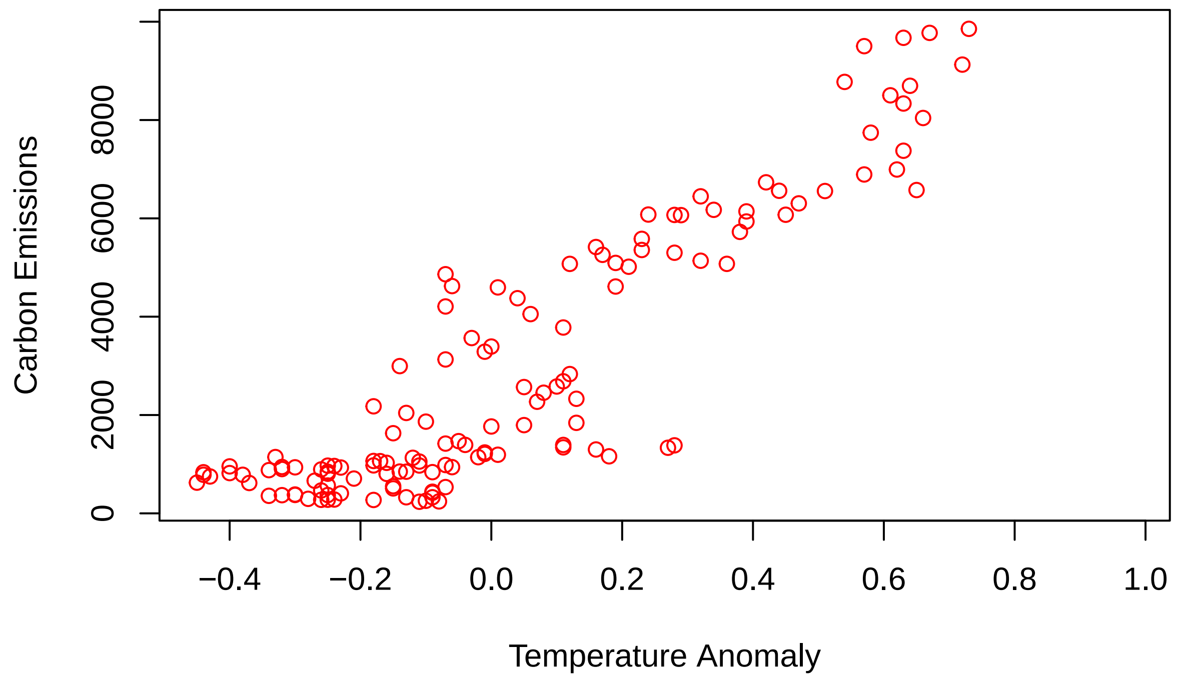
Using the `temp_carbon` dataset, select just those years with temp_anomaly values above the median value. Note, you may have to deal with "NA" values. I also recommend looking at the `temp_carbon` help documentation to find more info on the temp_anomaly variable.

```r
# Write your code here.
# There are several ways to do this.
temp_carbon <- temp_carbon[!is.na(temp_carbon$temp_anomaly),]
med <- median(temp_carbon$temp_anomaly)
filter(temp_carbon, temp_anomaly > med)
```

```
##    year temp_anomaly land_anomaly ocean_anomaly carbon_emissions
## 1  1937        -0.01        -0.02         -0.01             1209
## 2  1938        -0.02         0.17         -0.10             1142
## 3  1939         0.01         0.10         -0.03             1192
## 4  1940         0.16         0.07          0.20             1299
## 5  1941         0.27         0.10          0.35             1334
## 6  1942         0.11         0.06          0.13             1342
## 7  1943         0.11         0.07          0.12             1391
## 8  1944         0.28         0.19          0.32             1383
## 9  1945         0.18        -0.07          0.30             1160
## 10 1946        -0.01        -0.01         -0.01             1238
## 11 1951         0.00        -0.06          0.02             1767
## 12 1952         0.05        -0.05          0.08             1795
## 13 1953         0.13         0.20          0.10             1841
## 14 1957         0.07        -0.04          0.11             2270
## 15 1958         0.13         0.15          0.12             2330
## 16 1959         0.08         0.09          0.08             2454
## 17 1960         0.05         0.00          0.07             2569
## 18 1961         0.10         0.12          0.09             2580
## 19 1962         0.11         0.16          0.09             2686
## 20 1963         0.12         0.21          0.08             2833
## 21 1966        -0.01        -0.05          0.01             3288
## 22 1967         0.00         0.01         -0.01             3393
## 23 1969         0.11        -0.08          0.17             3780
## 24 1970         0.06         0.05          0.06             4053
## 25 1972         0.04        -0.17          0.11            4376
## 26 1973         0.19         0.34          0.14            4614
## 27 1975         0.01         0.14         -0.04            4596
## 28 1977         0.21         0.25          0.19            5016
## 29 1978         0.12         0.10          0.12            5074
## 30 1979         0.23         0.17          0.24            5357
## 31 1980         0.28         0.31          0.26            5301
## 32 1981         0.32         0.52          0.25            5138
## 33 1982         0.19         0.11          0.22            5094
## 34 1983         0.36         0.50          0.30            5075
## 35 1984         0.17         0.06          0.20            5258
## 36 1985         0.16         0.10          0.18            5417
## 37 1986         0.23         0.30          0.21            5583
## 38 1987         0.38         0.45          0.36            5725
## 39 1988         0.39         0.58          0.32            5936
## 40 1989         0.29         0.36          0.27            6066
## 41 1990         0.45         0.66          0.37            6074
## 42 1991         0.39         0.53          0.34            6142
## 43 1992         0.24         0.24          0.23            6078
## 44 1993         0.28         0.35          0.25            6070
## 45 1994         0.34         0.48          0.29            6174
## 46 1995         0.47         0.78          0.35            6305
## 47 1996         0.32         0.35          0.31            6448
## 48 1997         0.51         0.64          0.46            6556
## 49 1998         0.65         0.98          0.52            6576
## 50 1999         0.44         0.78          0.31            6561
## 51 2000         0.42         0.62          0.34            6733
## 52 2001         0.57         0.84          0.46            6893
## 53 2002         0.62         0.95          0.49            6994
## 54 2003         0.63         0.94          0.52            7376
## 55 2004         0.58         0.81          0.49            7743
## 56 2005         0.66         1.08          0.50            8042
## 57 2006         0.63         0.97          0.50            8336
## 58 2007         0.61         1.12          0.43            8503
## 59 2008         0.54         0.89          0.41            8776
## 60 2009         0.64         0.90          0.54            8697
## 61 2010         0.72         1.14          0.56            9128
## 62 2011         0.57         0.91          0.44            9503
## 63 2012         0.63         0.95          0.51            9673
## 64 2013         0.67         1.03          0.53            9773
## 65 2014         0.73         1.01          0.63            9855
## 66 2015         0.92         1.39          0.75              NA
## 67 2016         0.98         1.50          0.79              NA
## 68 2017         0.90         1.38          0.72              NA
## 69 2018         0.82         1.18          0.68              NA
```

Finally, let's take the chance to review some base plotting. Using the `temp_carbon` data, make a scatterplot of the `temp_anomaly` variable versus the `carbon_emissions` variable. Format the axis labels and try figuring out how to change the color of the points on the plot (any color you want is fine).

```r
# Write your code here.
with(temp_carbon, plot(temp_anomaly, carbon_emissions, col="red", xlab="Temperature Anomaly", ylab="Car
```



Describe in your own words any pattern(s) that you see in the plot.

A: Carbon-emissions and temp_anomaly have a postiive correlation. As one increases, the other increases too.

What is the unit of analysis for this dataset as it is currently constructed?

We are comparing the relationship between temp_anomaly (in degrees Celsisus) and carbon_emissions(in metric tons). According to the graph they have a positive correlation. Temp_anomaly are directly correlated to increasing carbon_emissions.

If you are interested in even more practice, work through the exercises in sections 4.2 and 4.4 of the textbook:

rafalab.dfci.harvard.edu/dsbook/r-basics.html#exercises-8

rafalab.dfci.harvard.edu/dsbook/r-basics.html#exercises-9